2008

Time synchronization in ad hoc wireless networks

Andrew Reid
Lehigh University

Reid, Andrew

Time Synchronization In Ad Hoc Wireless Networks

January 2009

Time Synchronization

In

Ad Hoc Wireless Networks

By

Andrew Reid

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical & Computer Engineering

Lehigh University

January 2009

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

12-04-2008

Date

Thesis Advisor

Chairperson of Department

Acknowledgement

I am greatly indebted to my thesis advisor Professor Shalinee Kishore with providing the opportunity to fulfill the main requirement for the degree in Masters of Science in Electrical Engineering while at the same time providing constant encouragement to purse one's dreams and goals. Much is said of the person who is capable of overcoming adversity to get to the top but to achieve ones goals on the way it sure helps when the people at the top extend a hand to pull one up. THANK YOU!

I would also like to thank my mother for always having confidence in me and providing moral support in many ways during my academic career. In her life, failure was never an option but she knew that there were many different paths to success and always had faith. LOVE YOU!

Lastly, I would like to express my sincere thanks to the Electrical & Computer Engineering Department. What I have received and experienced is incomparable and unforgettable!

Table of Contents

Table of Figures

Abstract

Time Synchronization
In
Ad Hoc Wireless Sensor Networks
by
Andrew Reid
Masters of Science in Electrical Engineering
Lehigh University, 2009
Professor Shalinee Kishore, Advisor

In real world applications of distributed systems, a node's local clock may shift over time and at different rates, causing the perceived time and period of each node to differ. During the fusion of nodes' data or any type of coordinated action synchronization of physical time is required. Hence, developing a suitable method of achieving time synchronization is necessary to implement sensor networks in a variety of applications, e.g. environmental monitoring, military & civilian reconnaissance, communication systems, etc.

New approaches to time synchronization can better support the broad range of ever evolving technology applications. One such approach to achieving time synchronization is by using consensus algorithms. These algorithms aim to reach synchronization by having a community of members, i.e. nodes, reach a general agreement based on each member's opinion.

A special type of consensus algorithm is the Distributed Consensus Time Synchronization Algorithm. Based on the theoretical framework developed by fellow graduate student Gang Xiong, this thesis aims to develop a system to analyze the practical implementation of this type of consensus algorithm. First, the basic concepts for consensus algorithms as it pertains to distributed consensus time synchronization is described. Then, a literature review of different approaches to time synchronization and the theoretical framework of Xiong's work are laid out and finally we describe the system model and results of the practical implementation of his distributed consensus time synchronization algorithm.

CHAPTER 1

Introduction

In real world applications of distributed communication systems, a node's local clock may drift over time and at different rates, causing the perceived time and period of each node to differ. As many physical and network layer protocols require synchronization between communicating nodes, a common view of time between all nodes is required and must be available over the life of the system. Furthermore, in such distributed systems and many applications, the unavailability of a node can result in disruption of connectivity with other nodes and decrease the quality of service of the network, e.g., its ability to achieve a common notion of time across all network nodes. Hence, developing a suitable method of achieving time synchronization is necessary to maintain a network's quality of service.

In this chapter, a special class of mobile ad-hoc networks, called wireless sensor networks, is presented; an abstract example of time synchronization, as well as the necessity of time synchronization in wireless sensor networks is described. I then give an overview and contribution of this thesis.

1.1 Wireless Sensor Networks

Tremendous advances in technology have enabled the deployment of inexpensive, small-sized, low-power terminals that can both conduct sensing (i.e., monitoring) functions and communicate with each other. Due to advances in the development of these terminals, wireless sensor networks have emerged over the past few years and received ongoing

attention from the research community. As a special class of mobile ad-hoc networks, wireless sensor networks can be formed spontaneously using available radio (e.g., spectrum, time, power) resources, without the need for any infrastructure. This lack of infrastructure requires that sensor nodes communicate directly with one another and use distributed or decentralized mechanisms to share communication resources with each other. For example, if a particular node's destination node is out of range then intermediate nodes must transmit (i.e., relay) each other's transmissions (e.g., packets) to the desired final destination. This results in a multi-hop communication network.

Although characterized as an ad hoc network, sensor networks experience several specific characteristics, e.g., limited energy sources (due to the small size of sensor nodes), high density roll out (for accurate sensing data collection), and cheap and unreliable sensor nodes (due to simplified, inexpensive node design).

Regardless of these limiting factors, in a variety of applications, sensor nodes are required to maintain accurate time synchronization, e.g. moving target object tracking, reconnaissance and surveillance, environmental monitoring, etc. [ref Gang' paper here] This necessitates network algorithms that achieve and maintain global time synchronization at all network nodes, i.e., distributed mechanisms to provide a common timescale for local clocks of nodes in the network.

1.2 Timing Synchronization Problem

Timing synchronization can be viewed as the following problem: A person, e.g., John Doe, is a normal commuter. John hates being late for work and wishes to always catch the train his wants. To achieve this, John obtains the reference time that is available at the train station and adjusts his watch to show the same time. Due to the imperfections in

John's watch, it does not measure time intervals accurately and therefore tends to drift away from the reference time at the train station. To maintain synchronization with the time at the train station, John must periodically synchronize his watch with the reference time. Using this example, we see that time synchronization is more accurate the more often a local clock (i.e., John's watch) is synchronized, i.e., synchronization is an ongoing process.

Since there is a sole resource for measuring time (the train station clock), John and any other person can achieve synchronization implicitly. But if that resource is unavailable, synchronization can only be achieved by communication between two or more entities, e.g.,, if John and Jane only have access to personal times and are able to communicate by some means. In such a scenario, synchronization can be achieve if John were to send Jane the reading on his watch, and Jane then adjusts her own watch based on either the time or the time offset between their clocks. Here too, periodic synchronization would be needed, since both John's and Jane's watches may drift at different rates, to ensure a specific synchronization error level.

1.3 Time Synchronization in Sensor Networks

Time Synchronization is a critical piece of infrastructure in any distributed system. In sensor networks, a multitude of factors make time synchronization important, while simultaneously making it more difficult to achieve than in traditional, i.e. wired, networks. Having a common view in time is a requirement of any entity to reason about events that occur in the physical world. For example, precise time is needed to measure time in the tracking of moving objects [1], low power TDMA radio scheduling [2], and environmental monitoring [3].

As can be seen from the many uses of time synchronization in sensor networks, it vital in the understanding of sensor information. But the broad spectrum of requirements, such as precision, lifetime, scope, and availability, is challenging in a distributed system and become particularly daunting when considering wireless sensor networks. For example, when nodes use large batteries they can run for long periods of time, while other systems have nodes that are constrained to "wake up" only when an event is triggered, after which it processes and send information, then returns to "sleep".

Regardless of these limiting factors, sensor nodes are required to maintain accurate time synchronization. This necessitates network algorithms that achieve and maintain global time synchronization at all network nodes, i.e., aims to provide a common timescale for local clocks of nodes in the network.

1.4 Overview

In this thesis, the theoretical base of the DCTS algorithm at the physical layer is presented in Chapter 2. The work discussed in said chapter is credited to Gang Xiong, fellow research assistant responsible for the analytical development. In Chapter 3, we present the main contribution of this thesis by examining the performance of the DCTS algorithm in practical wireless sensor network scenarios. Finally, conclusions and possible future work are addressed in Chapter 4.

CHAPTER 2

Literature Review

2.1 Introduction

This chapter sets up the theoretical framework, developed by fellow graduate student Gang Xiong, needed to understand the practical implementation presented in Chapter 3. But first, we begin with various methods of time synchronization is discussed and the basic concepts of consensus algorithms as it pertains to distributed consensus time synchronization is explored.

2.2 Metrics for Comparing Synchronization Algorithms

The main metrics used to measure the performance of a synchronization algorithm is variance and convergence time. The variance that is measured is that of the estimate that is produced. This is only true if the parameter being estimated is equal to the mean of the sample distribution. In the case of the Distributed Consensus Time Synchronization Algorithm, the estimate is biased, i.e. the estimator's expected value and the true value of the parameter being estimated is nonzero. Hence, the Minimum Mean Square Error (MMSE) is a more appropriate metric. The convergence time is the number of symbols (units of time) required to achieve that MMSE.

Synchronization Accuracy is another important parameter used in the evaluation of different time synchronization algorithms. It determines how often the timing needs to be re-estimated, i.e. how the time reference of each node needs to be synchronized again.

Although some classes of algorithms are described, no comparisons are made with the Distributed Consensus Time Synchronization Algorithm.

2.3 Time Synchronization Methods

Today, there are many methods to achieve time synchronization in distributed systems. Such systems include the U.S. Global Positioning System [] and radio stations operated by the National Institute of Standards and Technology [] provide references to standards for time and frequency.

In the absence of GPS to maintain accurate timing, wireless sensor nodes may rely on the presence of a *timing manager node* to align network nodes to a common notion of time. The resulting synchronization algorithms are inherently centralized in nature and are highly susceptible to failure of such manager nodes. Alternatively, wireless sensor nodes can rely on each other (much as John and Jane did on our earlier example) to achieve synchronization using *decentralized timing synchronization algorithms.*

In this section, review methods used to achieve time synchronization in a variety of network configurations.

2.3.1 The Network Time Protocol

The Network Time Protocol (NTP) was developed by Dave Mills of the University of Delaware. In today's Internet standards, the time system used is the Coordinated Universal Time (UTC). The Network Time Protocol is the protocol used for distributing the UTC by means of synchronizing the clocks of computers connected to an Internet network.

A hierarchical system called "clock strata" is used in the implementation of NTP. In a system with 'n' stratum levels, n being an integer, stratum 0 would be defined as the reference clock, consisting of GPS clocks, cesium or rubidium atomic clock, etc. Each subsequent clock would be defined by its distance away from the reference, stratum 1 being the closest and directly connected to the reference clocks. It should be noted that the reference clocks are not connected to the network but locally to the stratum 1 level of computers. The stratum 1 computers therefore provide the connection to the network of stratum 2, 3 ..., n-1 levels of computers. In other words, stratum 1 serves as the time server for the rest of the network. Stratum 2 computers send Network Time Protocol requests to the Stratum 1 time serves and Stratum 3 computers do the same to Stratum 2 computers. Stratum 3 computers can then themselves serve as time serves for lower strata computers in the network.

NTP is the now the standard for time synchronization on the internet. It provides the means to accurately correlate event logs between devices. Events that may include but not limited to tracking network usage, reduce confusion over shared file systems where modification of files need to be consistent, etc.

2.3.2 Master-Slave

The Master-Slave communication model establishes a master and slave relationship between two devices or processes. Once this relationship is established, the direction of control is always from the master to slave. Master-Slave clocks work with this model when attempting to perform time synchronization. Global Positioning Systems keep time very close to that of UTC timescale and therefore excellent candidates for master clocks.

2.3.3 Pulse Coupled Oscillators

In distributed synchronization, one strategy uses the behavior of pulse coupled oscillators, a term which refers to systems that oscillate periodically in time and interact each time they complete an oscillation. In [4], Mirollo and Strogatz study this spontaneous synchronization phenomenon and derive a theoretical framework based on pulse-coupled oscillators for the convergence of synchrony. Generally speaking, the way that these fireflies achieve synchronization is by initially flashing and then observing the flashing of their neighbors. If they see a neighbor flash, then they flash back in response and so on until all local fireflies are flashing at the same time. The pulse-coupled oscillator approach to network timing synchronization imitates this behavior by having network nodes initially transmit pulses to each other and then counting down to a subsequent similar transmission. If after transmitting a pulse a node receives a pulse from its neighbors, then it accelerates it's time to the next pulse transmission. By appropriately modeling the countdown mechanism and the acceleration in response to a pulse reception, this decentralized timing synchronization algorithm can be used to achieve the same reference of time (simultaneous pulse transmissions) amongst a collection of distributed nodes.

Using the theoretical results, [5] and [6] apply the theory to time synchronization of entities in wireless networks. As stated in [6], a pulse-coupled oscillator is completely described by its phase function $f(t)$. As this function linearly increases, it reaches some firing threshold denoted $f_{th}(t)$. If not coupled to any other oscillator, it periodically fires a pulse at the threshold level. However when coupled to others, oscillators are capable of receiving pulses. After receiving a pulse, the oscillator's phase is incremented by a phase

increment and fires its own pulse. This phase increment is defined by the Phase Response Curve (PRC). It is shown in [4], given the proper response curve, a network of pulse coupled oscillators always converges; in other words, all oscillators firing at the same time.

2.4 Distributed Consensus Time Synchronization

Another class of decentralized timing synchronization algorithms, and the focus of this work, is distributed consensus time synchronization (DCTS). In this approach, the DCTS algorithm relies on the local time information exchange between two or more nodes in a wireless sensor network. This exchange can occur using either MAC layer time-stamped packets or via physical layer pulse signals. In this thesis, the physical layer based time synchronization is considered and the wireless sensor network is modeled using undirected graph theory. As in the John and Jane time exchange model described earlier, each node in the DCTS approach modifies its current time based on a weighted average of the differences measured between the received times from its neighboring nodes and its own local time.

2.4.1 Defining Consensus

Formally stated, the meaning of consensus is a general agreement among members of a community which is based on each group member's opinion, but also the process of which that agreement is reached. Nodes in ad-hoc networks achieve consensus by reaching an agreement on a quantity of interest that depends on the state of all nodes [12], i.e. the local time present of each node.

The process behind consensus decision making begins with the topic under scrutiny []. In case of the wireless sensor network it is the local time of each individual node. From there, a proposal is presented to the community members and the members make a decision on if they agree with the proposal or not. If they agree then they have reached consensus but if not then each community member draws conclusions based on their opinions, i.e. based on a node's own local time. From there, a modification to the original proposal is then again presented to the community members. This process is repeated for a set period of time or until consensus is approved, either complete consensus among the community members or within some percentage.

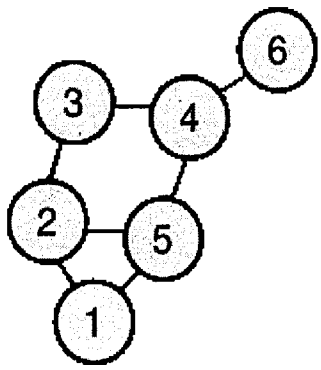
Consensus decision making aims to include as many of the community members to establish a consensus on any given topic. The more members involved in the decision making process results in a faster and more accurate decisions can be made.

2.4.2 Network Model via Graph Theory

The study of graphs is known as graph theory. The term "graph" used in this thesis is not to be confused with the common usage of the term "graph" to mean the plot of a function. In this thesis, the term "graph" is used to describe a network of points connected by lines. The points are more commonly referred to as graph vertices and the lines often referred to as graph edges.

Xiong's modeling of a wireless sensor network in [], [], & [], states, an undirected graph $G = (V, E)$, consisting of a set of n nodes $V = \{1, 2, \ldots, n\}$ and a set of edges E. Each edge is denoted as $e = (i, j) \in E$ where $i \in V$ and $e \in V$ are two nodes connected by edge e. We assume the presence of an edge (i, j) indicates that nodes i and j can communicate with each other reliably, i.e., all nodes j for which edge (i, j) is given.

From this network model, A is denoted as the adjacency matrix of G. The adjacency matrix of an undirected graph G with n vertices V results in an n x n matrix where the entry a_{ij} $(i \neq j)$ is the number of edges from the vertex i to the vertex j. In figure 2.1 (a), a simple example of an undirected graph with 6 vertices and 6 edges and figure 2.1 (b) is its corresponding adjacency matrix [13].



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

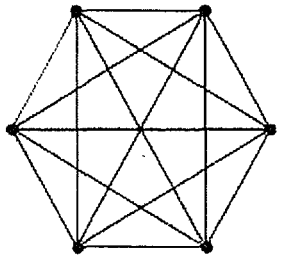
Fig. 2.1 (a) Undirected Graph (b) Adjacency Matrix

2.4.3 Network Topology

In this section, we detail the network topologies used during the analytical study of the DCTS algorithm. These topologies, although not ideal for practical sensor networks, are capable of providing valuable insights into the behavior of the algorithm.

Complete Network

A Complete Network is modeled as a graph in which each pair of graph vertices is connected by an edge, i.e. each node is capable of "communicating" with all other nodes in the network. A complete graph with n graph vertices is denoted as K_n.

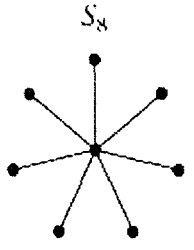


K_6

Fig. 2.2 Complete Graph

Star Network

A Star Network is modeled as a graph in which all *n-1* vertices each has only one edge and the remaining vertex has a total of *n-1* edges. Such a network is common in computer network topologies, where the node with a edge count of *n-1* is usually labeled as a central hub or switch for all the other nodes or computers on the network.



S_8

Fig. 2.3 Star Graph

Ring Network

A Ring Network is modeled as a graph in which there are *n* vertices each having two edges. A given i^{th} graph vertex yields edges $(i+j)^{th}$ and $(i-j)^{th}$ graph vertices, resulting in a circular path. In a network utilizing this network topology, the network is very susceptible to the malfunction of a node since a circular path creates the signal path. To avoid this problem, some ring networks add a "counter rotating ring" to form a type of redundant topology [11].

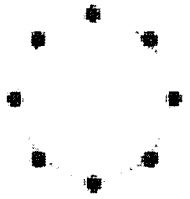


Fig. 2.4 Ring Graph

2.5 Distributed Consensus Algorithm

According to [9], the discrete nature of the algorithm can be thought of as an iterative process having Markovian properties. To think of this as a Markov chain is to say that "given the present state, future states are independent of the past states". Meaning, that a given present state can capture all the information need to generate a future state and is not dependent on the past states. Simply defined as

$$t(k+1) = t(k)H$$

where $H = I - \epsilon L$ and a small $\epsilon > 0$. Therefore, with an graph G with Laplacian L the theoretical definition of the first order DCTS algorithm is

$$\vec{t}(k+1) = (I_n - \varepsilon L)\vec{t}(k)$$

Xiong in [7], expands the DCTS algorithm to a theoretically *M-th* order DCTS algorithm:

$$\vec{t}(k+1) = (I_n - \varepsilon L)\vec{t}(k) - \varepsilon \sum_{m=1}^{M-1} c_m(-\gamma)^m L\vec{t}\,(k-m)$$

with the initial conditions $\vec{t}(-M+1) = \cdots = \vec{t}(-1) = \vec{t}(0) = \vec{z}$, where $\vec{z} = [z_1, z_2, \ldots, z_n]^{\mathrm{T}}$ and M is the order of the DCTS algorithm.

CHAPTER 3

Practical Implementation

3.1 Introduction

The main contribution of this paper is the practical implementation of the Distributed Consensus Average Algorithm and the results from that implementation. In this chapter, the simulation results of the consensus algorithm in the form of various plots. Those plots include the evolution of the Mean Square Error over a given iteration time as well as that of the Local Time information of the nodes over the iteration period, but first the practical implementation model is described.

3.2 Practical Implementation Model

From the previous chapter, the time update rule for the Distributed Consensus Time Synchronization Protocol is defined in [7] as:

$$t_i(k+1) = t_i(k) + \varepsilon \sum_{j \epsilon N_i} \left(t_j(k) - t_i(k)\right) + \varepsilon \sum_{m=1}^{M-1} \Delta t_i(m)$$

$$\Delta t_i(m) = c_m(-\gamma)^m \sum_{j \epsilon N_i} \Big(t_j(k-m) - t_i(k-m)\Big),$$

Assuming a 1st order update rule, i.e. M=1, then the practical model of the distributed consensus protocol evolves to:

$$t_i(k+1) = t_i(k) + \varepsilon \sum_{j \epsilon N_i} \alpha_{ij}\left(t_j(k) - t_i(k)\right)$$

where α_{ij} is defined as a weighted coefficient based on the received power. This is chosen because of the fact that the higher the received power the higher the reliability and therefore should have greater influence over the updating of node *i*'s local time. Similarly defined in [10], the weight is defined by:

$$\alpha_{ij} = \frac{P_{ij}}{\sum_{j \in N_i} P_{ij}}$$

This equation states that the weight between node *i* and node *j* is the received power of node *j*, P_{ij}, divided by the total received power experienced by node *i*, the receive node.

The above modification provides a practical time updated rule for a first order consensus time synchronization algorithm to be implemented with the following two steps:

1. Each node estimates the arrival time and power of the other neighbor nodes during the observation window.
2. Each node updates its own transmission time according to the modified update rule and then transmits the packet in the next time period T.

In this practical implementation, the packet only includes the training sequence used to detect the arrival of the other nodes, i.e. nodes $j \in N_i$. As Xiong mentions, we can choose the range of the window observed by the received node, node *I*, and it is chosen and shown in Figure 3.1.

By performing classical correlation to the received node's observation window, the arrival time of the other nodes can be estimated, assuming they appear in the window. Also, all received powers are compared against a threshold because is a received power is deemed too low make a significant enough contribution to the update of the receive nodes' local time then it is discarded.

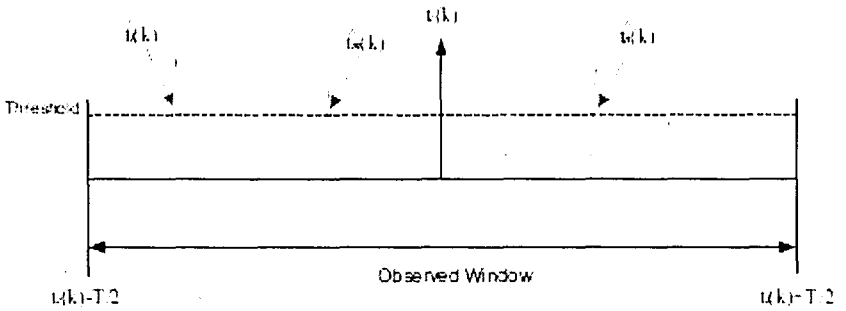


Figure 3.1 Observation Window

3.3 Structured Network Results

The first structured network used to observe the behavior of the consensus algorithm is that of the complete network. Figure 3.2 shows the evolution of the of the local time information of the nodes.

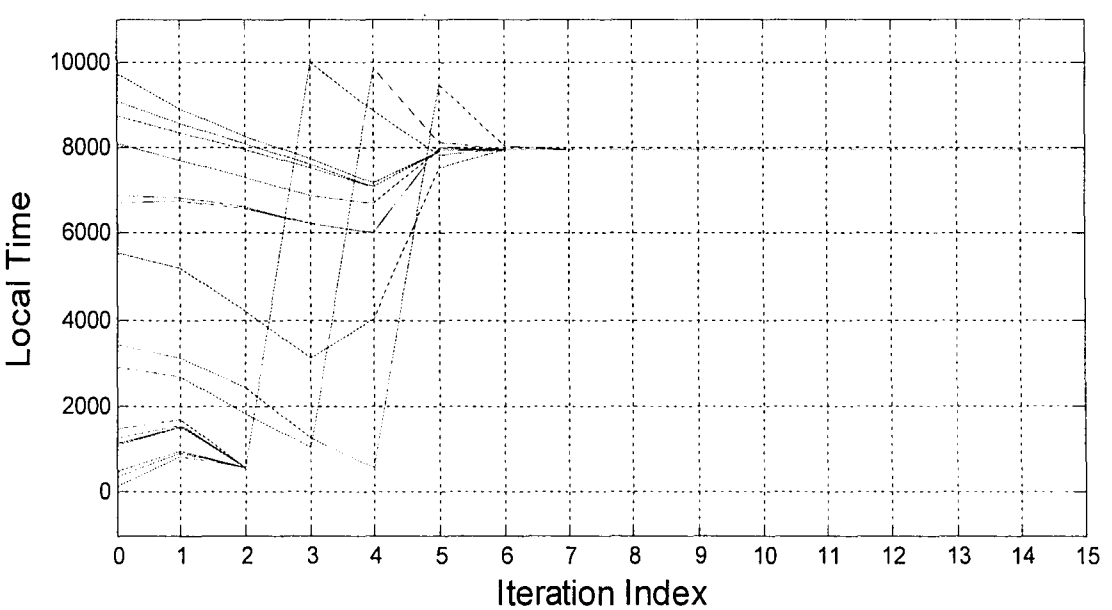


Figure 3.2 (a) Evolution of first order DCTS algorithm in Complete Network

Subsequently, the Mean Square Synchronization Error (MSSE) was also plotted.

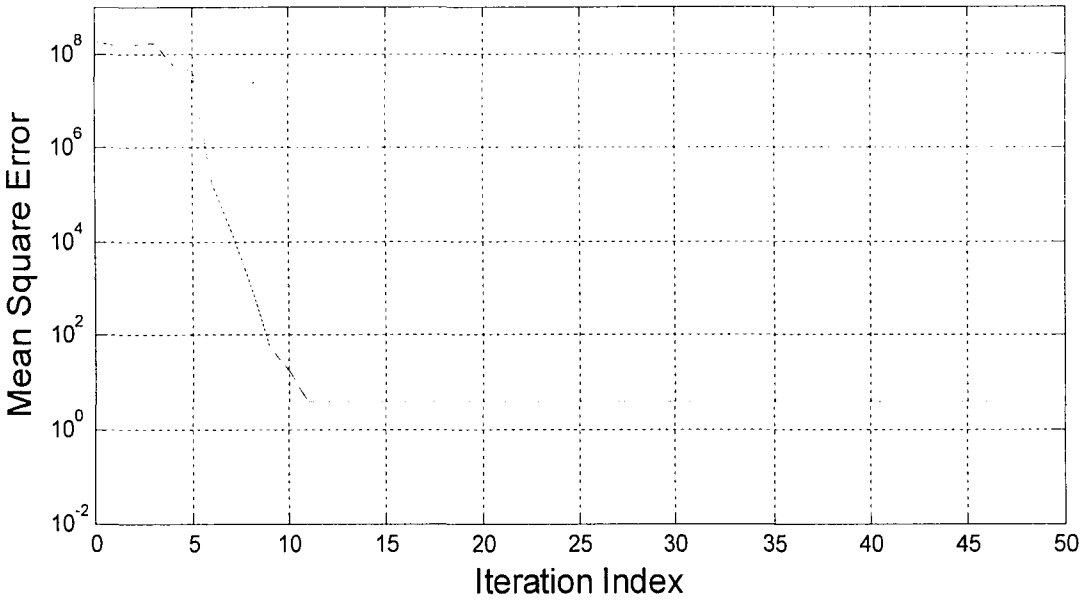


Figure 3.2 (b) MSSE of first order DCTS algorithm in Complete Network

As can be seen in the results of the other structured networks, the complete network converges faster and has a much smaller steady state MSSE than that of the Star and Ring Networks, respectively shown in Figure 3.3 (a) & (b) and Figure 3.4 (a) & (b).

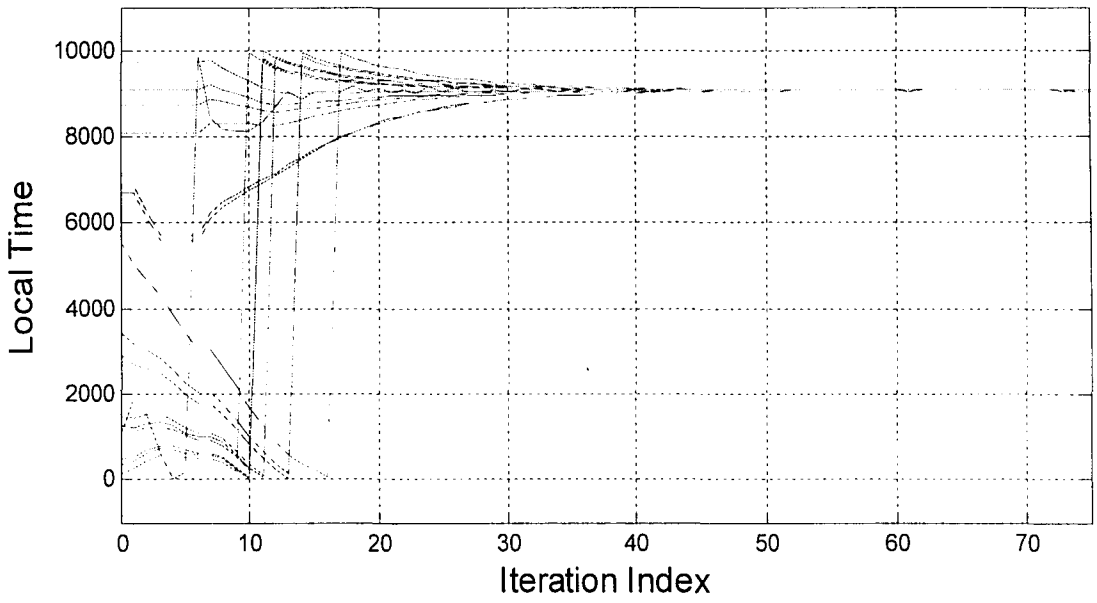


Figure 3.3 (a) Evolution of first order DCTS algorithm in Star Network

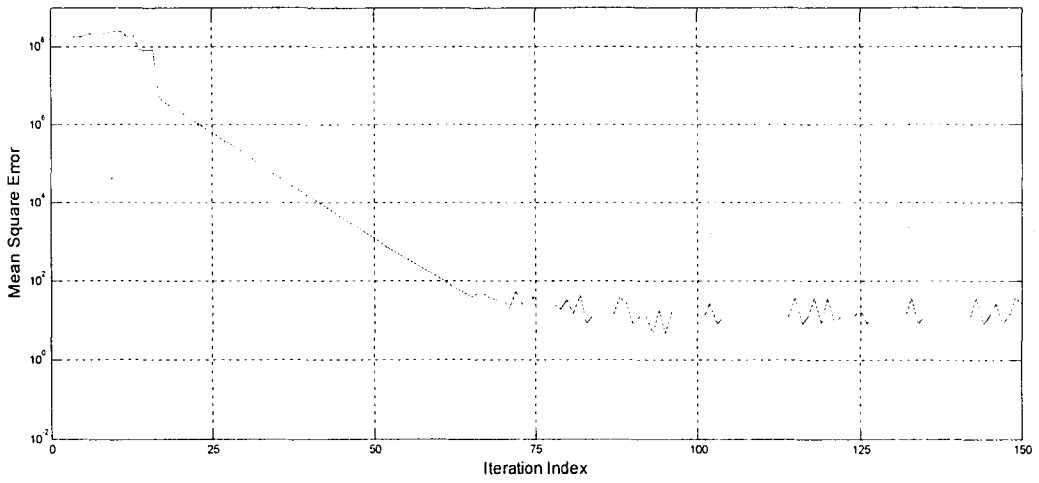


Figure 3.3 (b) MSSE of first order DCTS algorithm in Star Network

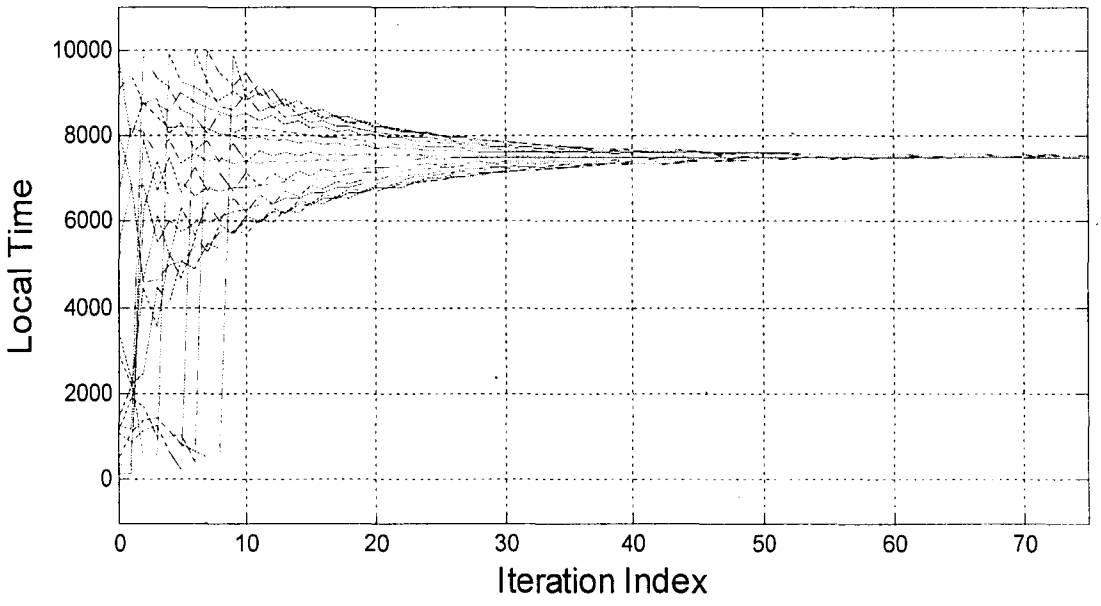


Figure 3.4 (a) Evolution of first order DCTS algorithm in Ring Network

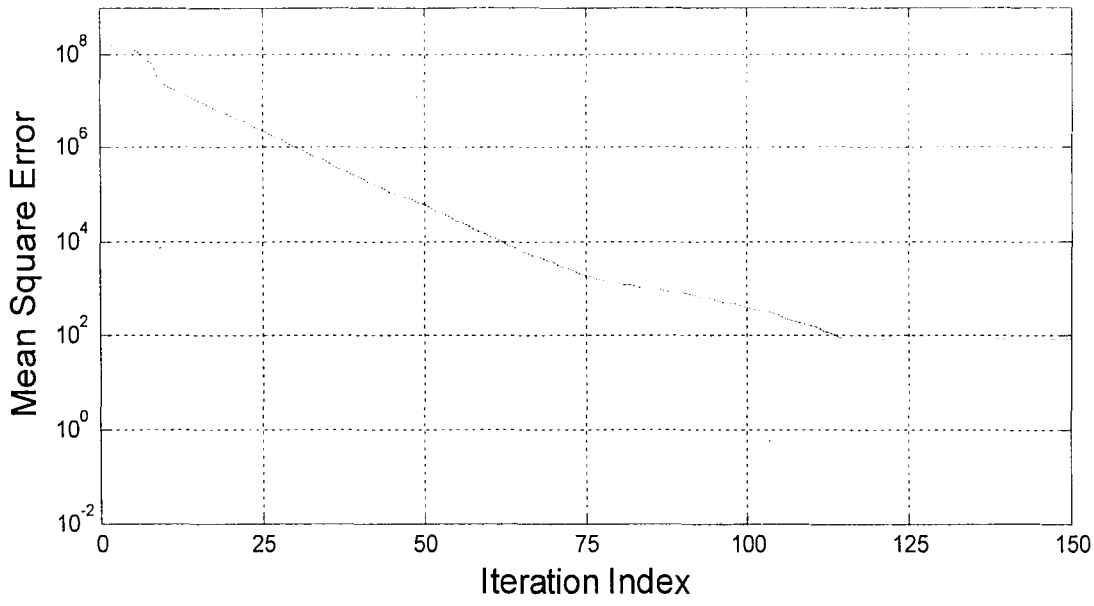


Figure 3.4 (b) MSSE of first order DCTS algorithm in Ring Network

3.4 Random Network Results

The random networks used to observe the behavior of the consensus algorithm in a small network, i.e. 16 nodes and 200 meter average distance between nodes. Figure 3.2 shows the evolution of the of the local time information of the nodes.

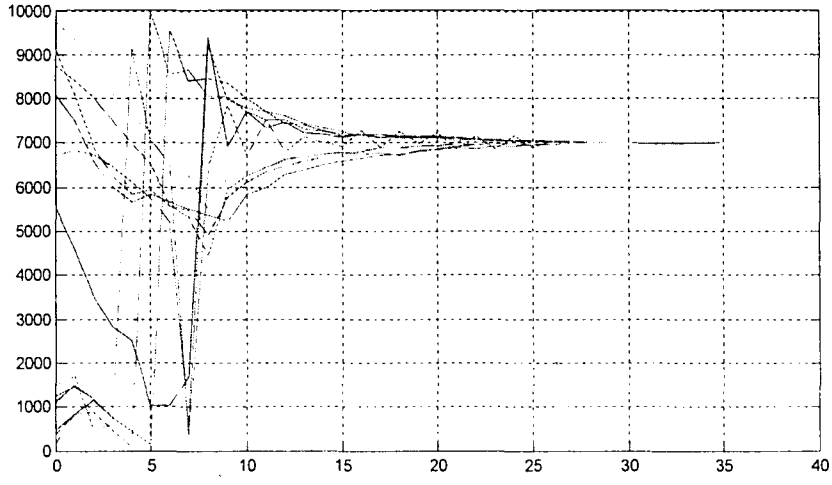


Figure 3.5 Evolution of first order DCTS algorithm in Random Network

CHAPTER 4

Conclusion & Future Work

4.1 Contributions

In this thesis, we present an overview of the Distributed Consensus Time Synchronization algorithm and show how it behaves in practical scenarios. In the results, 1 iteration was equivalent to that of 1 millisecond. Therefore we show that in a complete network, i.e. complete graph, that convergence can be achieved in less than 10 milliseconds. This is of course in the absence of arrival time, i.e. Propagation Delay. In [7], the results found in this thesis can be compared to the theoretical finds of Xiong, which would provide deeper understanding of the algorithm's behavior.

4.2 Future Work

Although from the results one can get a general perspective on the behavior of the DCTS algorithm in practical scenarios, much more work has to be done in order to fully understand the capability of the DCTS algorithm.

The higher order DCTS Algorithm is one example of how this work can be expanded. In [7], Xiong shows that one can experience gains in convergence time but using a high order DCTS algorithm.

Also, what other network topologies can be used to study the behavior of the algorithm. In graph theory, wealth of other network topologies exists. There is even the possibility of generating specific topologies to suit one's own application.

Lastly, it would be interesting to see how the DCTS Algorithm stacks up to other methods of time synchronization, pulse coupled oscillators, master-slave, NTP, etc.

Bibliography

[1] Wen-Chih Peng, Yu-Chee Tseng, Chih-Yu Lin. Efficient In-Network Moving Object Tracking in Wireless Sensor Networks. IEEE Transactions on Mobile Computing, 5(8): 1044-1056, Aug. 2006.

[2] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy, Wireless Integrated Network Sensors: Low Power Systems on a Chip. Proceedings of the European Solid State Circuits Conference, 1998.

[3] Hui Liu, Zhijun Meng, Shuanghu Cui. A wireless Sensor Network Prototype for Environmental Monitoring in Greenhouses. Wireless Communications, Networking and Mobile Computing, 2007, 2344-2347, Sept, 2007.

[4] Renato E. Mirollo, Steven H. Strogatz. Sychronization of Pulsed Coupled Boilogical Oscillators. Society for Industrial and Applied Mathematics, 50 (6): 1645-1662, Dec. 1990.

[5] Yao-Win Hong, Anna Scaglione. A Scalable Synchronization Protocol for Large Scale Sensor Networks and Its Applications. IEEE Journal in Communications, 23 (5): 1085-1099, May, 2005.

[6] Alexander Tyrrell, Gunther Auer, Christian Brettstetter. Firefly Synchronization in Ad Hoc Networks. 2006.

[7] Gang Xiong, Shalinee Kishore. High-Order Discrete Time Distributed Consensus Time Synchronization Algorithm for Wireless Sensor Networks.

[8] Gang Xiong. Practical Implementation for Global Time Over Wireless Sensor Networks. 2007

[9] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. Proceedings of the IEEE, 95 (1);215-233, Jan. 2007.

[10] E. Sourour and M. Nakagawa. Mutual Decentralized Synchronization for Intervehicle Communications. IEEE Transactions on Vehicular Technology, 48(6):2015-2027, Nov. 1999.

Vita

Andrew A. Reid is a graduate student in the Electrical & Engineering Department at Lehigh University, pursuing his Masters of Science in Electrical Engineering. He current holds a Bachelors of Science in Electrical Engineer (2006) with a minor in the Japanese Language from Lehigh University.

Before attending Lehigh, Andrew went to South Shore High School in Brooklyn, NY. Other than excelling academically, he was captain of the football team during his senior year and won the Coach Daniel Suhr Leadership Award.

During his undergraduate years, he has served as both Vice-President and President of the Lehigh University Chapter of the National Society of Black Engineers (NSBE). During that time he has won the NSBE Chapter Leadership Award, Dean of Students – Student Life Award, and the Student Senate Leadership Award.

Andrew's work experience includes working as a Research Assistant in the Electrical & Engineering Department at Lehigh University, 2005; Product Engineering Intern at CDG Technology, 2006; in Japan as a Design Engineering Intern at Yokogawa Electric Company, 2007; and as a Research Engineer at General Electric's Global Research Center, 2008. He has subsequently accepted an offer of full time employment at the Research Center in their Edison Engineering Leadership Development Program.

Andrew A. Reid was born in Mandeville, Jamaica on March 27, 1984 to Sylvena and Vernon Reid.

END OF TITLE